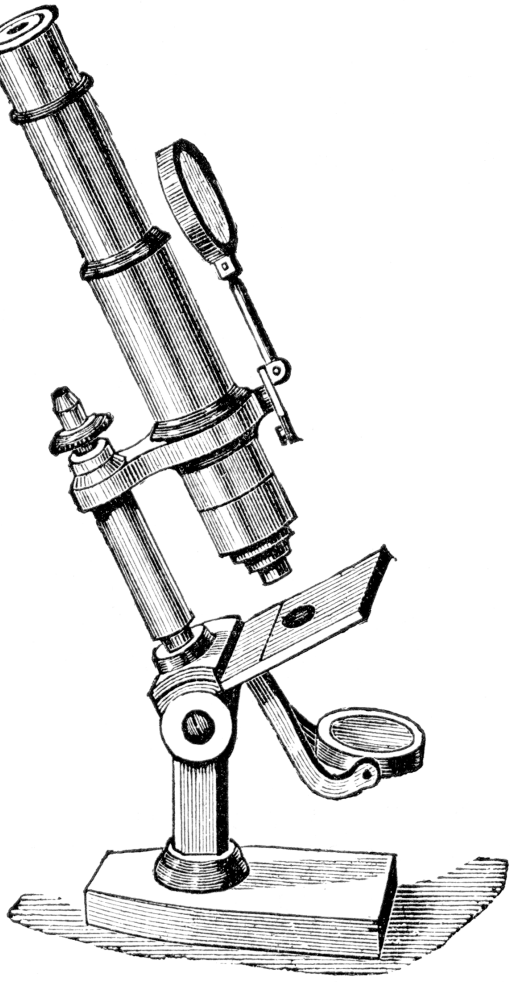


PENETRATION TEST REPORT

Evil Corp, Inc.

How I Hacked The Worst Company Ever!

2024-04-10

This report is for the sole information and use of Evil Corp, Inc.

Secure By Default, Inc.
+0 123 456 789
contact@sbd.local
https://sbd.local/

Executive Summary

Evil Corp, Inc. (i.e. the customer) engaged Secure By Default, Inc. to conduct a penetration test on their systems. In accordance with the customer the following types of penetration tests have been conducted:

- Add a new vulnerability rating system to the "report-generator"
- 5 Vulnerabilities Report

Vulnerabilities with severity scores between Track and Act (on a scale from Track–Act) have been found (see Table 1). A detailed list thereof, including descriptions on how they were found, and recommendations for mitigations, can be found in Section 2.

Table 1: The vulnerability classes, the highest Decision, and the number of issues per class.

Class	**Highest Decision**	**Issues**
Class Container	Attend	1
Class Deserialization	Track	1
Class Hijacking	Act	1
Class Injection	Attend	1
Class Phishing	Act	1
Total		**5**

My first taks was to implement a new flag on the "report-generator" tool. This flag (-s/–scoretype) should handle different Severity Score from **CVSS** which is the standard on this tool. I have made my own researchs about **CVSS** issues and why is not always ideal before find a different score type. **CVSS** is an industry standard maintained by NVD which define vulnerability as :

"A weakness in the computational logic (e.g., code) fouund in software and hardware components that, when exploited, results in a negative impact to confidentiality, integrity, or availability. Mitigation of the vulnerabilities in this context typically involves coding changes, but could also include specification changes or even specification deprecations (e.g., removal of affected protocols or functionality in their entirety)."

The point of scoring vulnerability is to prioritize patching process, obviously based on the *"danger"* (yes, this word needs more context to be defined but allow me to use it out of context for the sake of discussion) of each vulnerability. **CVSS is not a Risk Score** but a snapshot in time on how the vulnerability is in a specific software/platform/OS/ecc.. This create a small but focal gap to reflect reality of things, when a vulnerability is patched the CVSS score will decrease even if the patcher entity grade it as higher (see CVE-2023-35352 affecting Microsoft RDP, CVSS score of 6.5 but CRITICAL for Microsoft). Moreover is a quantitative approach used to prioritize the patching checklist (in the case of Microsoft RDP vuln, using CVSS would delay the fix even the high rate of threat) and cannot be ideal to client with lack of knowledge aout how CVSS score works, looking just at the numbers and create a leaderboard based solely on them. Lastly the context is not properly involved in the classification, every network is different and a CVSS of 9/10 in an enviroment could rely with proper detection/prevention technique or in an unimportant assets (in term of security/CIA triad) making the exploit of the vuln not that dangerous as the score rapresentation. With "context" we mean not just the victim environment but how much the vuln is exploited in the wild or existence of a public PoC, this should be considered into the evaluation to help clients to improve the security posture.

I have looked through some alternatives and I decide to use one which looks pretty interesting, this doesn't mean is ideal for the industry use but I wanted to implement something different and new. I decided to use SSVC (CVSS but the opposite way), a decision tree apporach which result is

a categorical variable called **Decision**. Stakeholder Specific Vulnerability Categorization leverage the vulnerability prioritizing methods involving stakeholders involved during th patching process, this new side introduction alone make it really different from the simple CVSS number. The results is based on decision tree which can be used for every gap found into the victim environment and should take care of the big picture not just the vulnerability itself. This would mark the same vulnerability inside *hMail* different for equal vuln inside *OutLook*, taking into account how the vulnerable assets is used inside the client *"mission"*. I used the CISA SSVC calculator because is simple and concise, the Decision Tree can be summarized like it follows:

- **State of Exploitation** = present state of the vuln (not future prediction); *None*, *PoC* or *Active*
- **Automable** = difficult and speed of exploitation and consequent actions; *Yes* or *No*
- **Techical Impact** = can be compared on CVSS severity score, how much control the Threat get after exploitation; *Partial* or *Total*
- **Mission & Well-Being** = the sum of 2 categorical variable (Mission Prevelance + Well-Being Impact); *Low*, *Medium* or *High*

The full information can be found here on how the decision tree work. Obviously different trees can be developed (because stakeholder change) so SSVC is not a one-shot solution itself and ad-hoc decision tree could be created for specific type of clients (private company, public institution, critical infrastructure, ecc..). The peculiarity of such system is the results at the end of the tree, a category not a number. It would not just prioritize but also suggest how to behave on every vulnerabiliy found, you could make a checklist and work concurently on every issues with an horizontal approach (with proper attention) for everyone inside the patching process:

- **Track** = No action but track the vulnerability situation. Consider to change your approach based on the new informations (for instance a public PoC or APT use)
- **Track+** = Like *Track* but monitor it closely and be prepared for future changes
- **Attend** = Actively request info about this vulnerability and assistance as well, notify (at least) internals about it giving the right attention. Actions should be done
- **Act** = Every stakeholder must be notified in order to meet and plan short-term mitigation/prevention technique.

The SSVC CISA score is similar to the CVSS with obviously no numbers but **Decison Value**, in the *issues files* have been saved like the following under the *severity section* :

```
Track
SSVCv2/E:P/A:Y/T:T/P:M/B:M/M:L/D:T
```

Regarding the challenge itself, working with categorical data forced me to make important addition (no copy-&-paste with parameter changes) on the whole repository (all python and latex files have been touched). I have explored the whole workflow of the script and learn some python usage new for me (Jinja2 was not that easy to understand at the start), I tried to change just the essential part of the repository to maintaint the identity of the codes and recognize immediately the changes and how they melt with pre-exisisting task. Same thing applied for the template and style of final PDF.

(Note that I have manually changed some SSVC string manually to have it all of the same length, this is because different path in the tree may return more value insidethe SVVC string so consider this as a beta. I worked under an assumption that a standardize tree have been implemented so all SSVC string have same length making copy-paste enough)

Contents

1 Introduction

Evil Corp, Inc. (i.e. the customer) engaged Secure By Default, Inc. to conduct a penetration test on their systems. In accordance with the customer the following types of penetration tests have been conducted:

- Add a new vulnerability rating system to the "report-generator"
- 5 Vulnerabilities Report

1.1 Personnel

The following people were involved in this penetration test:

- Alessio Stefan (alessiosteph@proton.me)

1.2 Scope

Between yyyy-mm-dd and yyyy-mm-dd the following components have been analyzed:

- `evil-corp.com` (application server)
- `https://4ever-evil.com/` (web application + API)

All test-related IP traffic originates from the following addresses:

- 192.168.0.0/24

1.3 Methodology

As asked by Patrick Himler I would grab 5 vulnerabilities found in my (little) CTF experience. This vulnerabilities would be managed as stand-alone so obviously the report sections are not connected to each others and are part of different machine. I would use the template to make a brief description of every vulnerability context so you can make an idea of the rating I gave to them, since I am not that keen with SSVC you could expect unaccurate results. I will explain briefly why I choose a specific path in the Decision tree to make it clear also I will try to use all the capabilities provided by the tools (image, link, snippet of code).

1.4 Requirements

Secure By Default required the following from the customer:

- technical contact personnel: fulfilled
- user accounts (at least two per type) for the web application: fulfilled
- API access token: fulfilled
- OpenAPI Specification document for the API: none provided

1.5 Provided Information

The customer has provided the following additional information:

- source code of the web application
- Apache config

1.6 Limitations

Due to the time-constrained nature of audits, it is common to encounter coverage limitations. The following limitations were identified during this engagement. Secure By Default recommends further review and/or retesting of the affected systems/components.

Dapibus

In dapibus est nec elit fermentum, et lacinia ipsum auctor. Praesent rhoncus, metus a tempor ultricies, ex ligula consequat elit, non sollicitudin massa libero vel odio. Duis blandit cursus metus. Quisque ac tincidunt lorem. Cras laoreet urna sed mi sagittis, et sagittis enim mattis. Praesent porttitor, dolor sagittis egestas hendrerit, ex neque sagittis sapien, vel tincidunt dui libero et sem. Fusce a odio faucibus, finibus augue non, aliquam augue.

Convallis

Nulla vitae nunc at magna molestie sodales convallis non libero. Pellentesque nec lectus lacinia, ultricies nulla in, laoreet lectus. Mauris eget urna efficitur, hendrerit diam eu, gravida mi. Phasellus porta lacinia ligula, ut dictum mi viverra at. Nullam dictum eros vitae massa ultrices, ac iaculis ante rhoncus. Donec pellentesque nec est sed porta.

1.7 Tools

The following tools have been used during the engagement:

- Nmap: network scanner
- cURL: command-line tool for transferring data using various network protocols
- Impacket-Suite: Set of tools for Active Directory enumeration/exploitation
- Burp Suite: web application security scanner
- ysoserial: Malicious serialization string crafter
- ffuf: Web Fuzzer written in Go
- Chisel: Port Forwarding tool taht support both TCP and UDP over HTTP
- Hydra: Cuncurrent bruteforcing tool supporting various protocols
- Metasploit: The most famous exploit framework you can find online! (it really needs presentation? lol)
- Responder: LLMNR/NBT-NS/mDNS Poisoner that support HTTP,SMB, webDAV and others protocol to use as rogue
- Hashcat: The most dangerous cat in the Kali Linux tools repository ()
- LaTeX: document preparation

2 Results

Table 2: Vulnerabilities and their severity (component “ANALYSIS”).

Vulnerability	**Decision**
[ANALYSIS] LDAP Injection	Attend

Table 3: Vulnerabilities and their severity (component “POV”).

Vulnerability	**Decision**
[POV] Insecure Deserialization	Track

Table 4: Vulnerabilities and their severity (component “PANDORA”).

Vulnerability	**Decision**
[PANDORA] Path Hijacking	Act

Table 5: Vulnerabilities and their severity (component “RUNNER”).

Vulnerability	**Decision**
[RUNNER] Container Administration Commands	Attend

Table 6: Vulnerabilities and their severity (component “PREMONITION”).

Vulnerability	**Decision**
[PREMONITION] Phishing	Act

2.1 [ANALYSIS] LDAP Injection

Similar to SQL or OS injection, LDAP injection use the protocol syntax into a text form (or similar) to retrieve data stored, commonly, in Active Directory enviroment. This could be really usefull for attackers who don't have valid credentials to enumerate usernames, password, descriptions and other LDAP field. In Active Directory we can read every data inside LDAP by design if we have credentials but with this vulnerability in play threat can circumnavigate this requirement. It become more dangerous with public facing services where non-authenticated actors could interact freely.

Context

The context is inside the first part of **Analysis** box, thanks to a nmap scan I was able to identify the presence of the AD domain called *analysis.htb* and the public machine was the Domain Controller (*dc-analysis.htb*) with a public webapp on port 80. With some basic enumeration/fuzzing I found the endpoint **/users** and, more in depth, **/users/list.php**. Both located into the subdomain **internal.analysis.htb** My intuition drove me to add some parameters until the **?name=** worked properly, with the "*" character I got a table with a username as output. Here I was 100% sure that LDAP injection attack vector was present (pretty obvious because the same fields in the table are the ones within the AD user catalog) {1}. After exploitation I was able to login in the website backend leveraging my unautenticated situation and move on until foothold thanks to unsafe File Upload that permits PHP reverse shell to be loaded server-side.

2.1.1 Evidence

2.1.1.1 internal.analysis.htb/users/list.php?name=*

The user I get on this website endpoint was **technician**, using **ASREP-roasting** (*impacket-GetNPUsers* tool) I was able to check if was a valid user inside the AD domain or not, luckly this was the case {2}. At first I tought was SQL because the "*" works the same with both LDAP and SQL but it requir a few tries to understand was LDAP for sure. I had to go AFK and review back LDAP syntax, it required 5/10 minutes and the attack flow looks clear enough to move on. The server side will just add a parenthesis and doesn't sanitize in any way, changing the parameter with `*)(description=*` returned the same output, we can enumerate the description of the user ***technician*** with adding characters before the last "". I tried manually the first character to be sure the attack could work and after reciving the output with `*)(description=9*` I know the description start with the character "9". The next step is obvious, we need to automate this process in order tofind the full string (is not uncommon to see plaintext passwords inside the "description" field because the lack of knowledge about who can see what in Active Directory). This is the python script :

```
import string
import request

finish = false
keys = string.ascii_letters + "#$%&'()+,-./:;<->?@[\\]^_{|}*"
url = 'http://internal.analysis.htb/users/list.php?name='
payload = ''
result = ''

while not finish:
    for i in keys:
        print(f'\r{result}{i}', end="")
        payload = result + i
        attack = url + f"*)(description={payload}*"
        resp = requests.get(attack)
        if "technician" in resp.text:
            result = result + i
            break
    finish = False


print(result)
```

The results after firing the script is "97NTtl*4QP96Bv", this password is not valid for Active Directory Domain but works for the backend login form (located at **/employees/login.php**). Here we can upload (and then review) file indiscriminately {3}, like a PHP reverse shell that allowed me to get foothold on the Domain Controller {4}.

EXTRA : Before login inside the backend I performed password spraying with the usernames found inside and the password retrieved from the script. The user "cwilliams" used the same pasword which allowed RPC access (but somehow not WinRM). {5}

SSVC Decision

- **Exploitation = PoC**, because *"private evidence of exploitation is tested but not shared"* is true thanks to the costum python script and is a well know attack
- **Automable = Yes**, the steps 1 to 4 of the kill-chain are fully automable
- **Technical Impact = Partial**, we retrieved just a specific field of the LDAP records about a single user. Exploiting this doesn't automatically gives you full controll of the vulnerable component
- **Mission & Well-Being = High**, the component is directly connected with the client mission (remember this machine is a domain controller)
- **Decision = Attend**, the sum of the previous choices

2.1.2 Affected Assets

- internal.analysis.htb (port 80)

2.1.3 Severity

Track Track* Attend Act

Exploitation

[PoC] Active None Physical

Automable

[Yes] No

Technical Impact

[Partial] Total

Mission and Well-Being

Low Medium [High]

Decision

Track Track* [Attend] Act

2.1.4 Recommendations

- Using LDAP in this tasks is uneccesary risk (especially for a webapp), find alternatives to avoid to make the webapp directly linked with LDAP data also take care of similar solutions inside your environment
- Regardless the origin of data implement sanitization of peculiar character (")" for instance) to avoid crafted payload to be injected
- Since the upload form needs to send a file directly to the SoC (and PHP file should be allowed) there is no need to stored in the webserver. Moreover favor private channel rather than backend form to sen this type of files
- Don't use the description field for hardcoded password and avoid password reuse inside your company. The whole password policy needs to be re-designed

2.1.5 References

- My full walkthrough of Analysis machine

2.1.6 Images

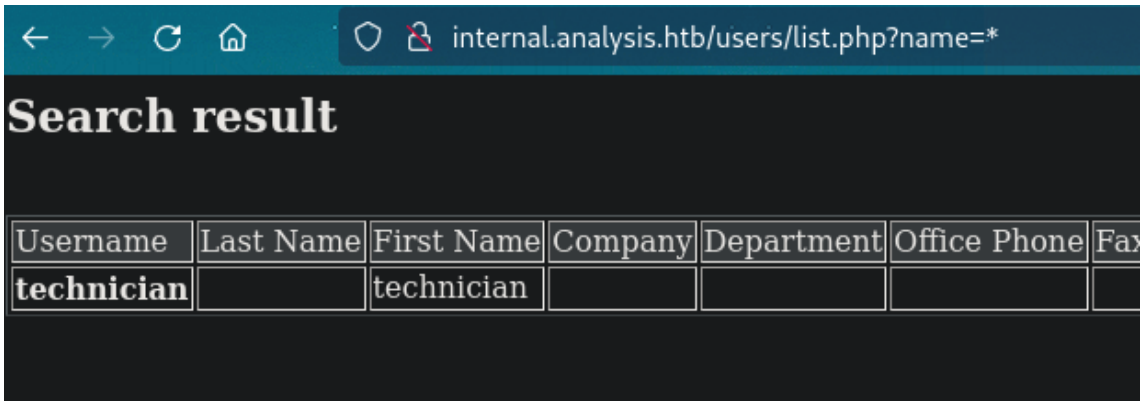


Figure 1: Output using the "*" as parameter of "name" variable

Figure 2: Check if user "technician" is a valid account inside the AD domain

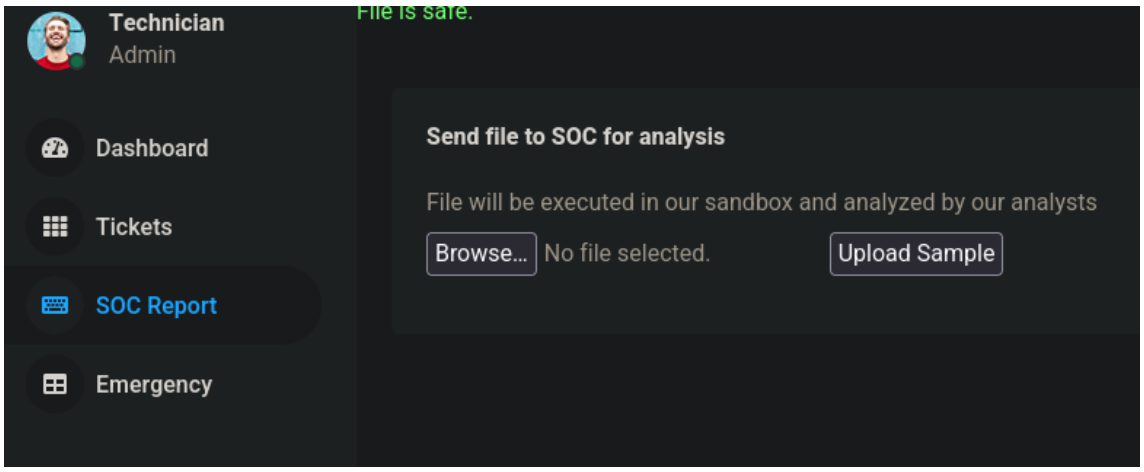


Figure 3: Upload form within the backend

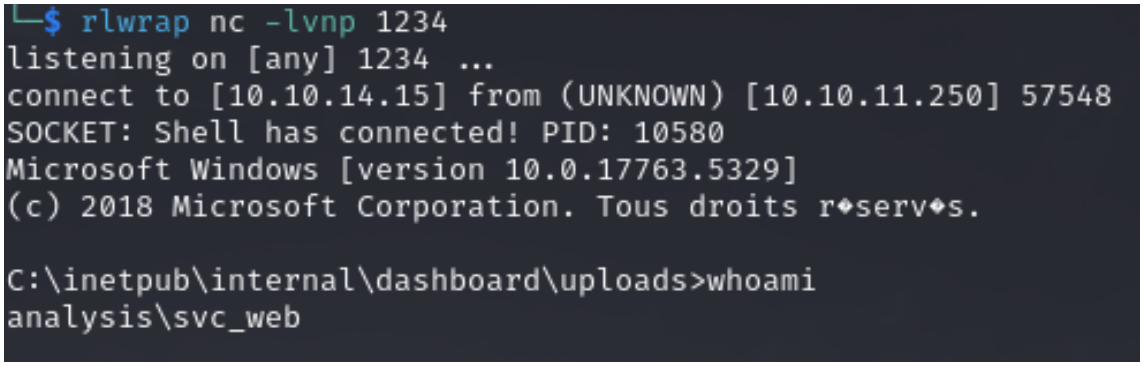


Figure 4: Reverse shell inside the domain controller

Figure 5: Password Spraying

2.2 [POV] Insecure Deserialization

Deserialization process take as input a format structured data and rebuilding into an object (eg:/ JSON, XML). This process is potential harmful when the deserialization string is user-controllable, manipulate the structured data could lead to command or object injection making the desrialization process the attack vector. Attackers needs to know the type of deserialization (eg:/ PHP, Java) and craft the serialized object exploiting known gadget that would be handled and executed by the victim machine (similar to ROP chain). The common impact of this vulnerability is RCE.

Context

Here I will cover the first part of the machine called POV, in order to exploit an ASPX insecure deserialization you need to find the decryption and validation keys through a LFI. With this 2 requirements I was able to craft the malicious ASP.NET serialized object that would be injected into the ViewState POST parameter to get a meterpeter shell.

2.2.1 Evidence

2.2.1.1 dev.pov.htb

During reconnaissance it has been found that the machine `pov.htb` have only the HTTP service (port 80) open and available to the public. The base domain is a static brochoure of the website where no interaction can be achieved. Subdomain bruteforcing (with **ffuf** tool) found `dev.pov.htb` which look interesting compared to the base domain. Here we have an overview of the (potential) backend technology thanks of a CV which can be downloaded through the endpoint `/portfolio` {6}. Looking closer the request with BURP the CV file is fetched through a POST parameter (`file`), I played with this changing the value and discover an LFI and RFI was possible. Knowing this I fired up Responder and use the LFI to fetch a file on the rogue SMB server in order to grab username and NTLMv2 Hash {7} {8}. The hash was uncrackable but at least we have harvested a valid machine username, plus inspecting the request highlight the presence of **ViewState** POST parameter. **ViewState** is used for session authenticaion (similar to cookies) and is well known that is vulnerable to **Insecure Deserialization** when is user controllable but we need more info about the validation/decryption key/alghorithm if we want to properly inject it without throwing an error. Without covering all the trial-&-error I made using the LFI pointing to `../web.config` uncover all the data we need to go further {9}. I create the ASPX malicious object using ysoserial with the following flags :

- **-p = ViewState**
- **-g TextFormattingRunProperties**
- **–decryptionalg="AES"**
- **–decryptionkey={AES_KEY}**
- **–validationalg="SHA1"**
- **–validationkey={SHA1_KEY}**
- **–path=/portfolio/contact.aspx** (can be whatever ASPX page)
- **-c {payload for web_delivery metasploit module}**

Finally we have crafted our own ViewState object and is ready to be injected {10}, now we can make a new request and change the original ViewState with the crafted one, after that a meterpeter shell as **web_svc** user will popout {11}. Foothold has been achieved!

SSVC Decision

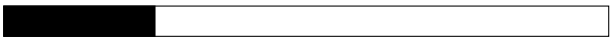
- **Exploitation = PoC**, because *"the vulnerability has a well known method of exploitation"*
- **Automable = Yes**, we automatically achieved RCE with the successful exploitation of the vulnerability

- **Technical Impact = Partial**, we are spawning as low privileges user so our potential damage is reduced and need additional steps to go further
- **Mission & Well-Being = Medium**, the vulnerable component is a simple workstation with no esential component for the client mission
- **Decision = Track**, the sum of the previous choices

2.2.2 Affected Assets

- dev.pov.htb (port 80)

2.2.3 Severity



Track Track* Attend Act

Exploitation

[PoC] Active None Physical

Automable

[Yes] No

Technical Impact

[Partial] Total

Mission and Well-Being

Low [Medium] High

Decision

[Track] Track* Attend Act

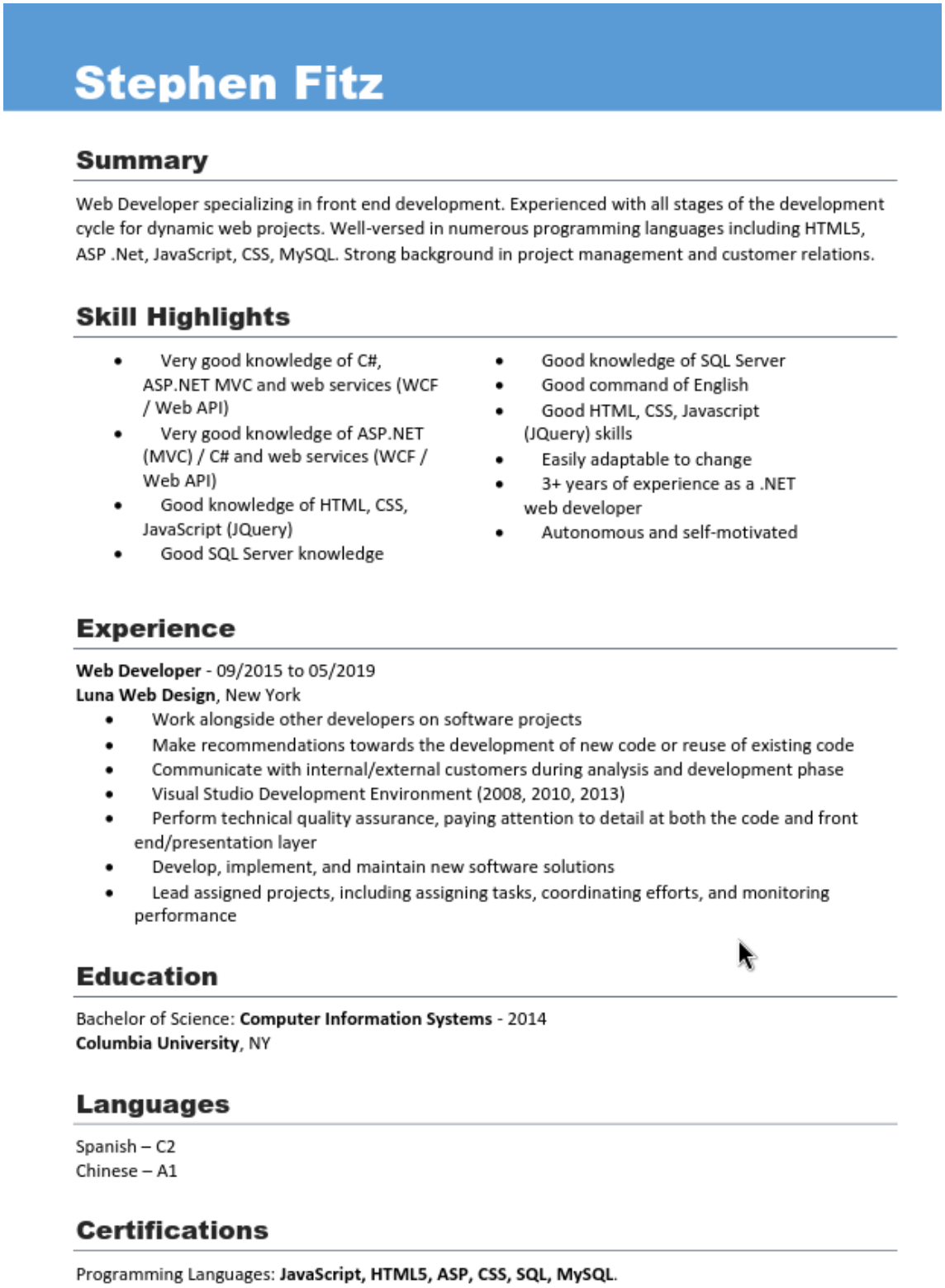
2.2.4 Recommendations

- The LFI is the key factor that allowed insecure deserialization to occur. Instead of using a parameter to fetch the file, create ad-hoc function to grab just the CV without the need of user-controllable parameters
- Never ever deserialize untrusted data, it would not be that simple as it sound but is a risk with relevant consequence
- Check the library used for serialization/deserialization process in order to find some flags/options to prevent code execution
- Removing/Disabling specific gadget can create a cascade effect on the whole infrastructure generating issues and errors. Do it but at your own risk and make proper testing, future implementation could suffer from that so is not suggested if not strictly needed

2.2.5 References

- My full walkthrough of POV machine

2.2.6 Images



Stephen Fitz

Summary

Web Developer specializing in front end development. Experienced with all stages of the development cycle for dynamic web projects. Well-versed in numerous programming languages including HTML5, ASP .Net, JavaScript, CSS, MySQL. Strong background in project management and customer relations.

Skill Highlights

- Very good knowledge of C#, ASP.NET MVC and web services (WCF / Web API)
- Very good knowledge of ASP.NET (MVC) / C# and web services (WCF / Web API)
- Good knowledge of HTML, CSS, JavaScript (JQuery)
- Good SQL Server knowledge
- Good knowledge of SQL Server
- Good command of English
- Good HTML, CSS, Javascript (JQuery) skills
- Easily adaptable to change
- 3+ years of experience as a .NET web developer
- Autonomous and self-motivated

Experience

Web Developer - 09/2015 to 05/2019
Luna Web Design, New York

- Work alongside other developers on software projects
- Make recommendations towards the development of new code or reuse of existing code
- Communicate with internal/external customers during analysis and development phase
- Visual Studio Development Environment (2008, 2010, 2013)
- Perform technical quality assurance, paying attention to detail at both the code and front end/presentation layer
- Develop, implement, and maintain new software solutions
- Lead assigned projects, including assigning tasks, coordinating efforts, and monitoring performance

Education

Bachelor of Science: **Computer Information Systems** - 2014
Columbia University, NY

Languages

Spanish – C2
Chinese – A1

Certifications

Programming Languages: **JavaScript, HTML5, ASP, CSS, SQL, MySQL**.

Figure 6: CV found into dev.pov.htb

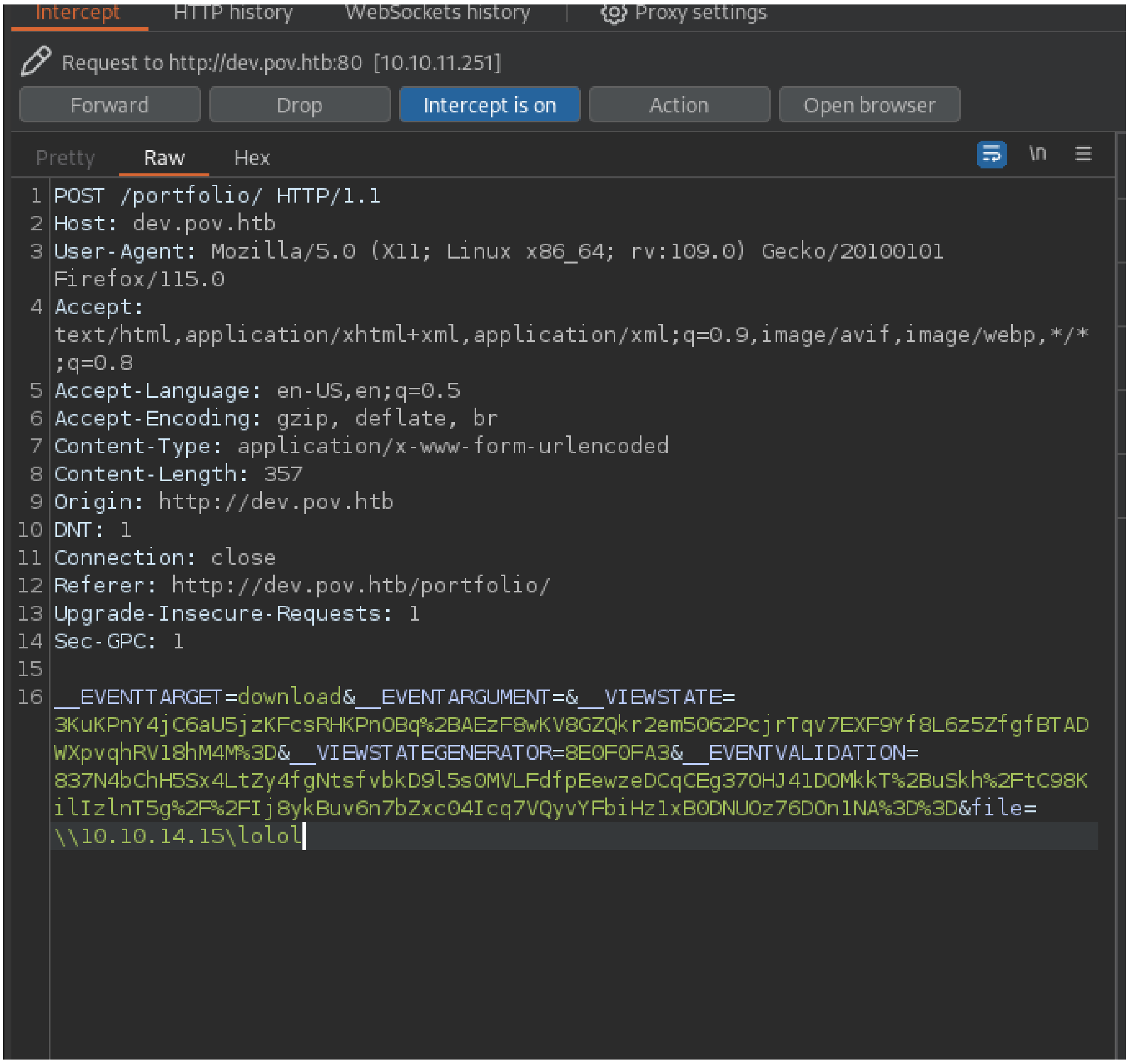


Figure 7: LFI/RFI using Burp Suite

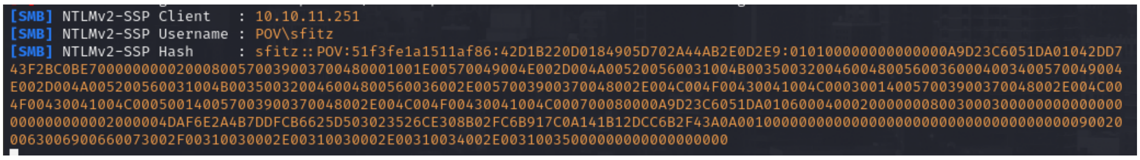


Figure 8: NTLMv2 Hash grabbing with Responder

```
Content-Length: 866

<configuration>
  <system.web>
    <customErrors mode="On" defaultRedirect="default.aspx" />
    <httpRuntime targetFramework="4.5" />
    <machineKey decryption="AES"
decryptionKey="74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80
D5E822F347183B43" validation="SHA1"
validationKey="5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1AC
FA1EDE22213BECEB55BA3CF576813C3301FCB07018E605E7B7872EEACE791AA
D71A267BC16633468" />
  </system.web>
    <system.webServer>
        <httpErrors>
            <remove statusCode="403" subStatusCode="-1" />
            <error statusCode="403" prefixLanguageFilePath=""
path="http://dev.pov.htb:8080/portfolio"
responseMode="Redirect" />
        </httpErrors>
        <httpRedirect enabled="true"
destination="http://dev.pov.htb/portfolio"
exactDestination="false" childOnly="true" />
    </system.webServer>
</configuration>
```

Figure 9: Content of the web.conig file discovered through the LFI

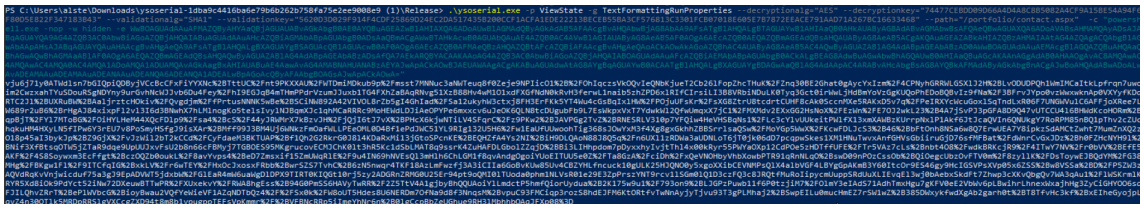


Figure 10: Crafting of malicious ViewState string

Figure 11: Meterpeter reverse shell

2.3 [PANDORA] Path Hijacking

The use of relative path inside codes could lead to Path Hijacking, the attacker change the $PATH variable in order to execute malicious costumversion of program instead the originals one. This doesn't target a specific programming language the only requirement is to call binaries or program at OS level. The attack is defined under the "Hijack Execution Flow" definition where an attacker abuse a pre-exisisting code in order to perform unwanted action. Is one of the basic methods of Privilege Escalation especially in Unix system where costum program with SUID/SUDO permissions allow to get root from low privilege users.

Context

Here we will cover the privilege escalation section of th machine **Pandora**, I get foothold through a well-known exploit of a vulnerable version of PandoraFMS (`v7.0NG.742`) which is a SQL injection that can be leverage to RCE on the machine. I got access as `matt` user and found a SUID program which is vulnerable to Path Hijacking to get RCE as `root`.

2.3.1 Evidence

2.3.1.1 pandora

With simple enumeration it didn't take much before I discovered a costum SUID binary called **"pandora_backup"** which is a simple backup tool for PandoraFMS {12}. Without downloading locally the file and do reverse engineering I tried to use `strings` linux utility to put an eye of the binary content, luckly it was not that complex and I see the gap that permits exploitation through Path Hijacking. The binary call `tar` but without the absolute path and without forcing a standard PATH variable inside the code {13}. I moved into `/dev/shm` and create a version of `tar` that just spawn a bash process, change the PATH enviroment variable giving precedence to this directory with `EXPORT PATH=/dev/shm:$PATH` and than reuse `pandora_backup`. This time it will not finish because when it hit `tar` command will spawn a shell as root thanks to the SUID {14}.

SSVC Decision

- **Exploitation = Active**, Hijacking Execution flow has been one of the most used technique is 2019-2021 even if is reduced is use is one of the favorite tricks by threat in the wild
- **Automable = No**, the steps from 1 to 4 are not fully automable and requires manual enumeration to discover the vulnerability (like reverse engineering)
- **Technical Impact = Total**, not just RCE has been achieved but high-level RCE because the SUID that permits to impersonate root user
- **Mission & Well-Being = High**, the vulnerable component is not a simple webserver but host the Pandora FMS monitoring software which can lead to remote control of other machines inside the network
- **Decision = Act**, the sum of the previous choices

2.3.2 Affected Assets

- pandora machine

2.3.3 Severity

Exploitation

PoC [Active] None Physical

Automable

Yes [No]

Technical Impact

Partial [Total]

Mission and Well-Being

Low Medium [High]

Decision

Track Track* Attend [Act]

2.3.4 Recommendations

- Be aware when assign SUID or SUDO permissions to softwares, in this case a backup could be done by a service account or an high-privilege profile which is not root.
- Costum binaries/tool should be full reviewed trying to avoid relative path and add an hard-coded PATH enviroment variable value to guard from Path Hijacking technique, same thing for DLL in Windows
- In general implement the least-privilege principle in every aspects of your infrastructure, if is not a developer account it don't need to cahnge PATH enviroment variable

2.3.5 References

- My full walkthrough of Pandora machine

2.3.6 Images

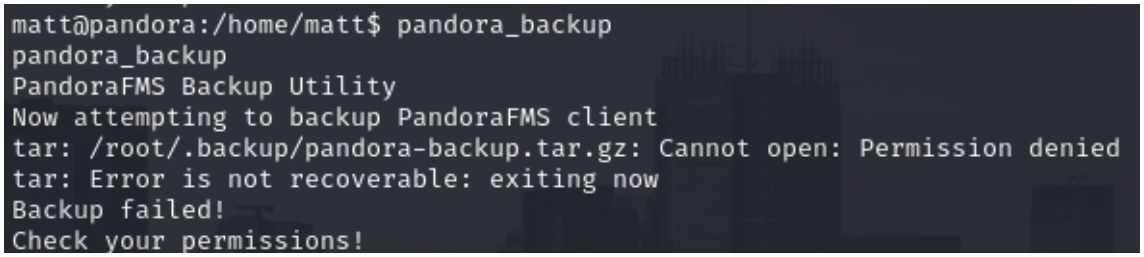


Figure 12: A low-priv run of pandora_backupbinary

Figure 13: Strings of pandora$_b$*ackupfile*

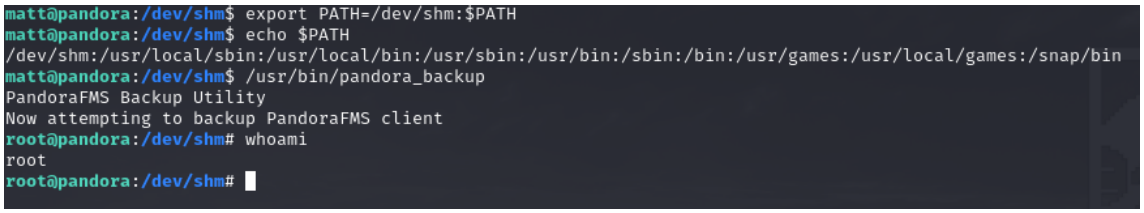


Figure 14: Exploitation through Path Hijacking

2.4 [RUNNER] Container Administration Commands

When adversary put their hands into container administration/orchestration can turn in dangerous consequence. There are various technique for both RCE and privilege escalation (and more) involving Kubernetes or Docker and companies should be aware of it. From entrypoint script to container escape makes this technologies a double-edge blade that needs to be handled carefully. Not to mention the flexibility and portability of containers could be exploited for horizontal movement or pivoting.

Context

Runner machine is a medium machine where the user flag should be grabbed through the exploit of CVE-2024-27918 affecting some TeamCity versions that allows to remotely create admin users without credentials. After login inside TC a private RSA key has been found and used to SSH inside the machine as `john`. Here Apache configuration file uncover a subdomain called `portainer-administration.runner.htb` hosting **portainer.io** docker orchestration software. From here we can deploy pre-existing image into the machine and abusing mount-binding I was able to retrieve the contents of `/root` folder and grab the flag.

2.4.1 Evidence

2.4.1.1 portainer-administration.runner.htb

After enumerating the machine internally I visited the **portainer.io** subdomain which is shielded by the login form {15}. With Hydra I was able to retrieve the passowrd of one of the users inside the machine, the valid credentials are `matthew:piper123`. Portainer.io is an UI orchestrator for Docker, the trick here is to translate every UI actions to `docker-compose` flag but looking at the documentation was not that hard. There is justs one predefined image that we can use (a simple `ubuntu:latest`), my first idea was to create a container instance where can reach a path reachable by the normal user, set a SUID bash as root (in the container) and than execute it in the bare-metal machine. I had some issue with this option (which is still possible) and decide to abuse **mount-binding** option to link the `/root` directory to another directory inside the container which, as root, contents can be fetched freely. First I need to create a **tmpfs storage** and set it as bind mount {16}. Save the storage and go in the deployment phase, set the storage volume just created and map it to a directory inside the container of your choice {17} and deploy the container into the machine. When ready, we can finally read the content of the `/root` directory and read the final flag! {18}.

SSVC Decision

- **Exploitation = PoC**, Portainer.io is really helpfull for the attacker and can modularize the same exploit on different machine
- **Automable = No**, you need credentials for both the local machine and portainer.io (also not all accounts can perform all the actions), moreover port forwarding is required and the port can change in different networks making hard the creation of a single automation code for this vulnerability
- **Technical Impact = Total**, in this specific case we can set malicious containers in every machine into the network (in this case the network is a single machine) permitting to move horizontally, vertically and pivoting into other network
- **Mission & Well-Being = High**, container are used for different tasks inside the client network and a sabotage of 1+ container can buff threat capabilities
- **Decision = Act**, the sum of the previous choices

2.4.2 Affected Assets

- portainer-administration.runner.htb

2.4.3 Severity

Track Track* Attend Act

Exploitation

[PoC] Active None Physical

Automable

Yes [No]

Technical Impact

Partial [Total]

Mission and Well-Being

Low Medium [High]

Decision

Track Track* [Attend] Act

2.4.4 Recommendations

- Check the password policy and implement bruteforcing/logon attempt count mitigation technique even within internal services, a simple solution that can make the attackers life harder
- Monitor properly creation/changes/deployment of new container inside your network and prepare incident response process in case of exploitation attempt
- Container relationships with other network components are complex and a small change could destroy your line of defense, never rely on a single security boundary when protecting from such threat. Meet all the stakeholders involved with containers (users,clients,developers,ecc..) to reach multiple solutions reaching the needs of everyone

2.4.5 References

- My full walkthrough of Runner machine

2.4.6 Images

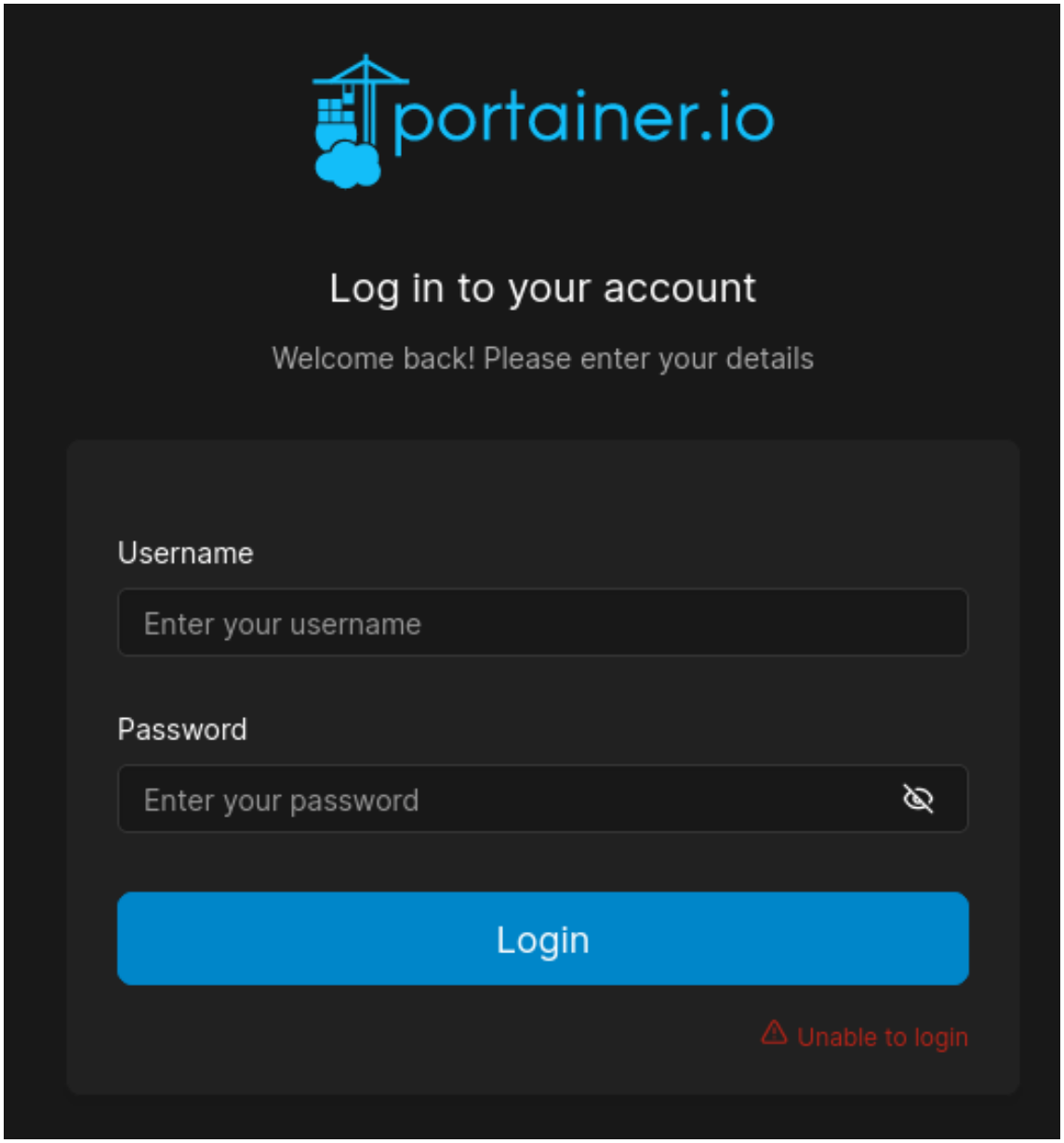


Figure 15: Portainer.io subdomain

Figure 16: Portainer.io Bind Mount creation

Figure 17: Portainer.io storage link with /root flag

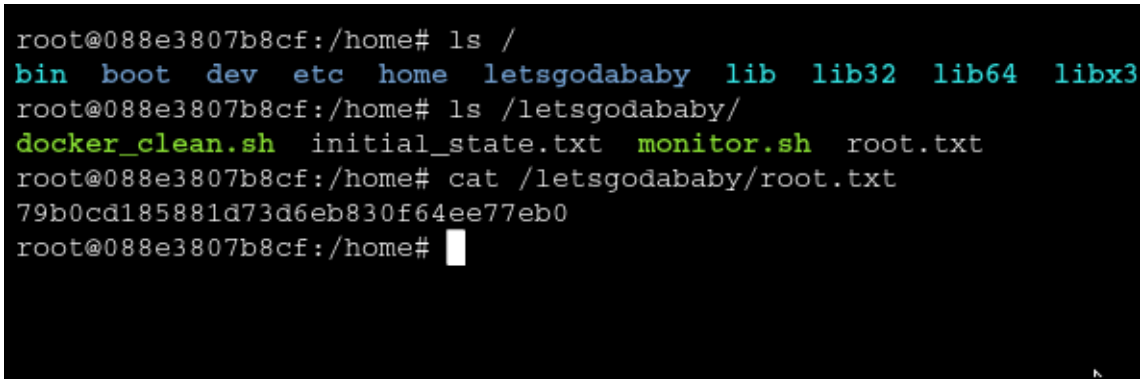


Figure 18: Root flag of RUNNER machine read through the container just created

2.5 [PREMONITION] Phishing

Phishing needs no presentation is an attack vector known even by non-technical people and is still one of the favorite TTPs by APT. There are different way to perform it : VBA script, HTA, and XXL are just a few examples that are used for the attack-chain. The general attack flow is the same, we try to convince the user to download/view/use a malicious attachment delivered in specific way, camouflate as legittimate and automate the malicious actions everything executed with the right spice of social engineering. Obviously is not simple at it seems other invisible threat (like AV) needs to be handled as well but in this final mini-report I wanted to cover the first part of the **Zephyr ProLab** (HackTheBox) where a simple but effective phishing method has been used to get initial access inside the victim network

Context

Zephyr first flag (*The Premonition*) start with just a linux webserver that rely on a full AD infrastructure, through this webserver we can upload PDF for the "Vacancies" section, using a crafted PDF I was able to retrieve NTLMv2 hashes changing the **/AA** field and pointing it to Responder SMB. The real difference with "traditional phishing" (where a sort of reverse shell would occur or fake login form setup) is that we steal silently without direct user action, just opening the PDF is necessary.

2.5.1 Evidence

2.5.1.1 painters.htb

Enumeration of **painters.htb** website (the hostname of the machine is *mail.painters.htb* so my creativity would assume is also a mailserver for the sake of the challenge) ended with the discovery of three different Job Apply form {19} each of them allow to upload a PDF and send it for review {20}. A PDF file can be really threatening when but only with the right information and weaponization. Hacking is not just about tools and techniques (remember what we said about users in the introduction?) but use of simple logic can clean up our head and under- standing. The "staff" would unlikely being some IT guy but maybe someone from HR or non-informatic specialist and we expect this type of professionals to log into SSH on a web server and use the bash to retrieve our PDF? Probably this PDF would be uploaded on the Linux machine and than passed somewhere else, we in an AD lab so 99% of chance that this "else" is going to be a Windows machine. With some research I discover the BADPDF module in metasploit which would create (or change) a PDF tuning the **/AA** field. this field contains 3 different variable:

- **/S** = The type of action to perform, we would use the GoToR (*Go To Remote*) which would tell the PDF editor the following variable needs to be fetched outside the document (outside means everything, even the machine itself)
- **/F** = This is the key point, here we can specify a remote file which in our case should be our malicious SMB server *\\[IP]\[FILE]*
- **/D** = The location of the fetched document to extract, not relevant for out purposes

The PDF creation is made really simple thanks to metasploit and after generating it we should go and upload on every upload form available to grab as much hash as possible. After some minutes Responder will start to flood {21}, that's great! Our intuition was right, the PDF has been opened on a Windows machine! using **hashcat** I cracked the Riley hash and get the plaintext (*P@ssw0rd*) and used on the SSH port of the mail/web server, luckly the password policy was not strong enough and allowed to reuse password in this linux machine and grab the first flag! {22}. Later on I have set up this machine as pivot to connect on the internal network and start playing with AD protocols moving further and further.

SSVC Decision

- **Exploitation = Active**, Phishing is still one of the most used technique in the wild, even with job boards or online forms (look what happened to Europol in these days, lol)
- **Automable = Yes**, Weaponization, delivery and exploitation are fully automable

- **Technical Impact = Partial**, retrieving user hash is cool but doesn't automatically means that we have full control on the vulnerable component (the webserver)
- **Mission & Well-Being = High**, in this specific case we get controll of the web and mail server with a single stone. These 2 components are foundamental for the security and mission posture of the client
- **Decision = Act**, the sum of the previous choices

2.5.2 Affected Assets

- mail.painters.htb

2.5.3 Severity

Track Track* Attend Act

Exploitation

PoC [Active] None Physical

Automable

[Yes] No

Technical Impact

[Partial] Total

Mission and Well-Being

Low Medium [High]

Decision

Track Track* Attend [Act]

2.5.4 Recommendations

- Is the third time in this paper, password policy needs to be reviewed from the start. No need to use the same passwords into 2 different machine especially if one of them is an important component of the infrastructure
- Zero-Trust approach on everything that originate from the outside, use detection room or sandbox when you need to open unknown files.
- Take care of outgoing and upcoming connection from untrusted source, if there is no needs that a machine inside AD needs to access a public facing SMB use the firewall to kill this connections (same thing applied to similar connections)

2.5.5 References

- The ProLab is under paywall and public walkthrough cannot be pubblished until the lab is active. I have attached on the email my full walkthrough in orderto be reviewed without breaking HTB labs

2.5.6 Images

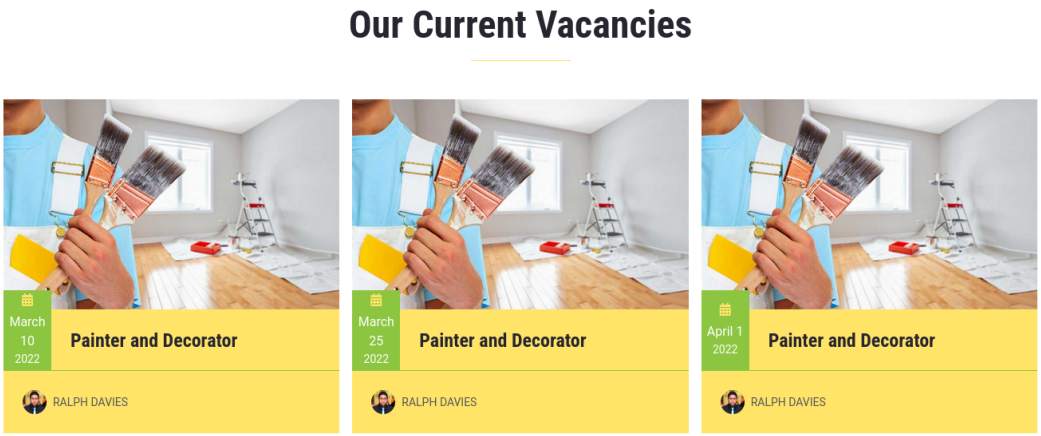


Figure 19: The 3 vacancies sections inside the website

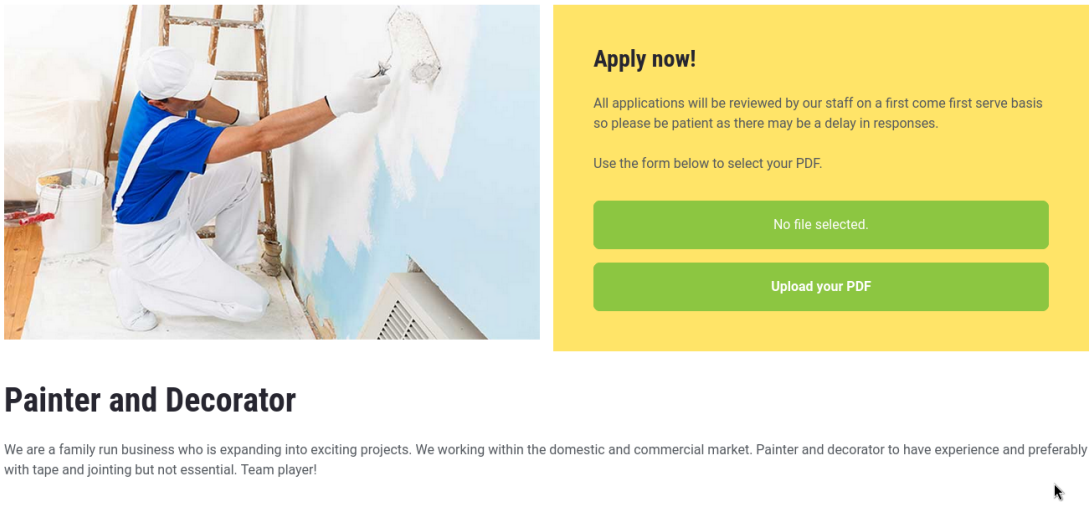


Figure 20: PDF upload form

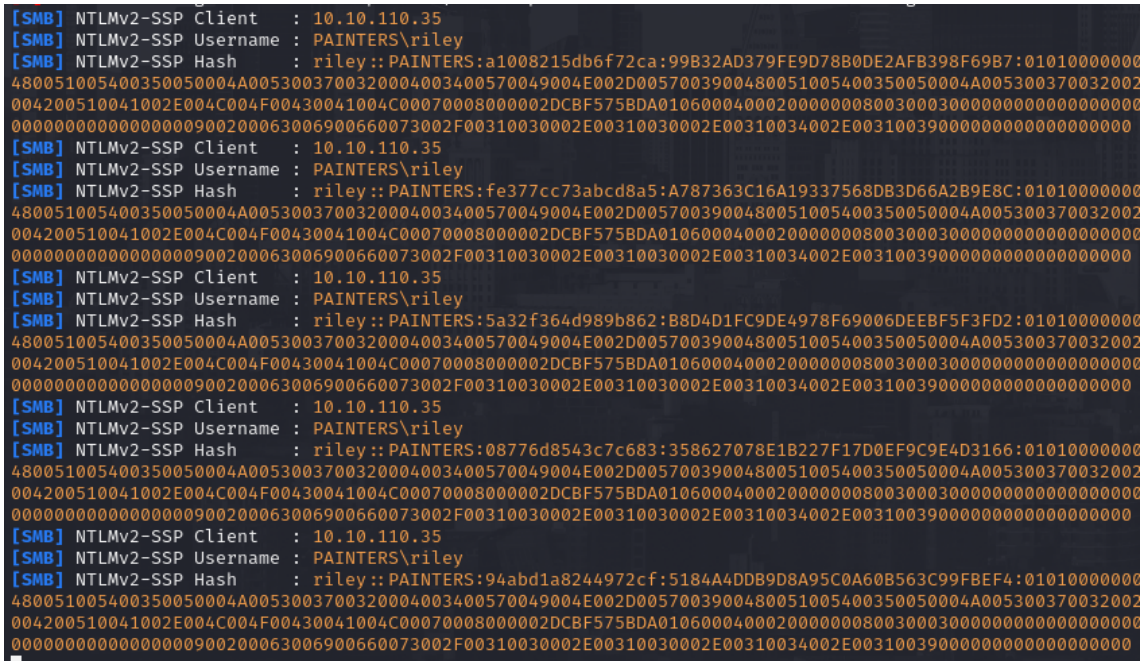


Figure 21: Responder output

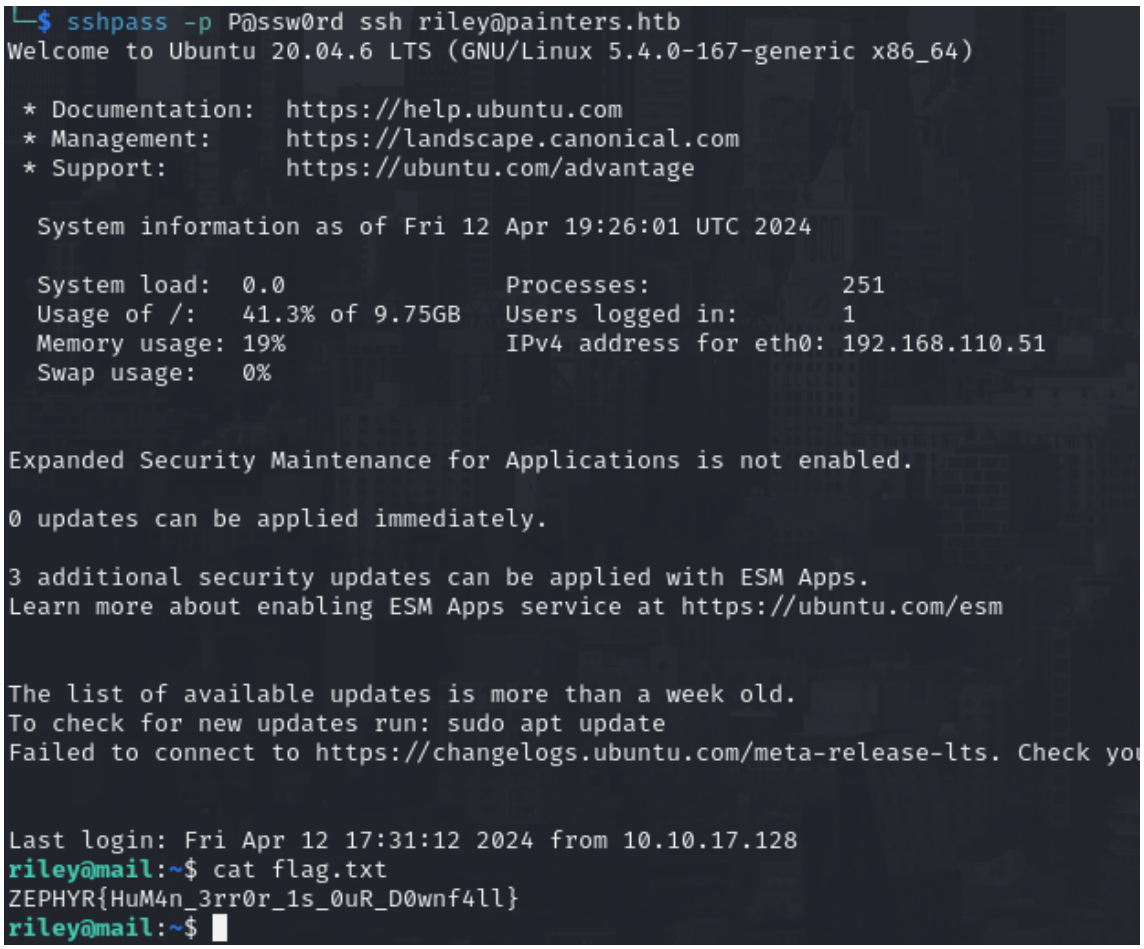


Figure 22: SSH and flag grabbing into the webserver